Java Deep-Dive Interfaces and Collections

A Day in the Life of a Full Stack Developer

One of the scrum team members is not available for a week. However, the team has committed that they will complete the functionality of authenticating their users. Pair programming with a senior developer will help Joe complete his task sooner. Joe agrees to pick this task to challenge his skill set. To help him complete the task sooner, a senior developer decides to assist him wherever required.

Joe has to develop a functionality where a user email address should be validated from a list of email addresses of users who have already signed up on their website.

In this lesson, we will learn how to solve this real-world scenario to help Joe complete his task effectively and quickly.

Learning Objectives

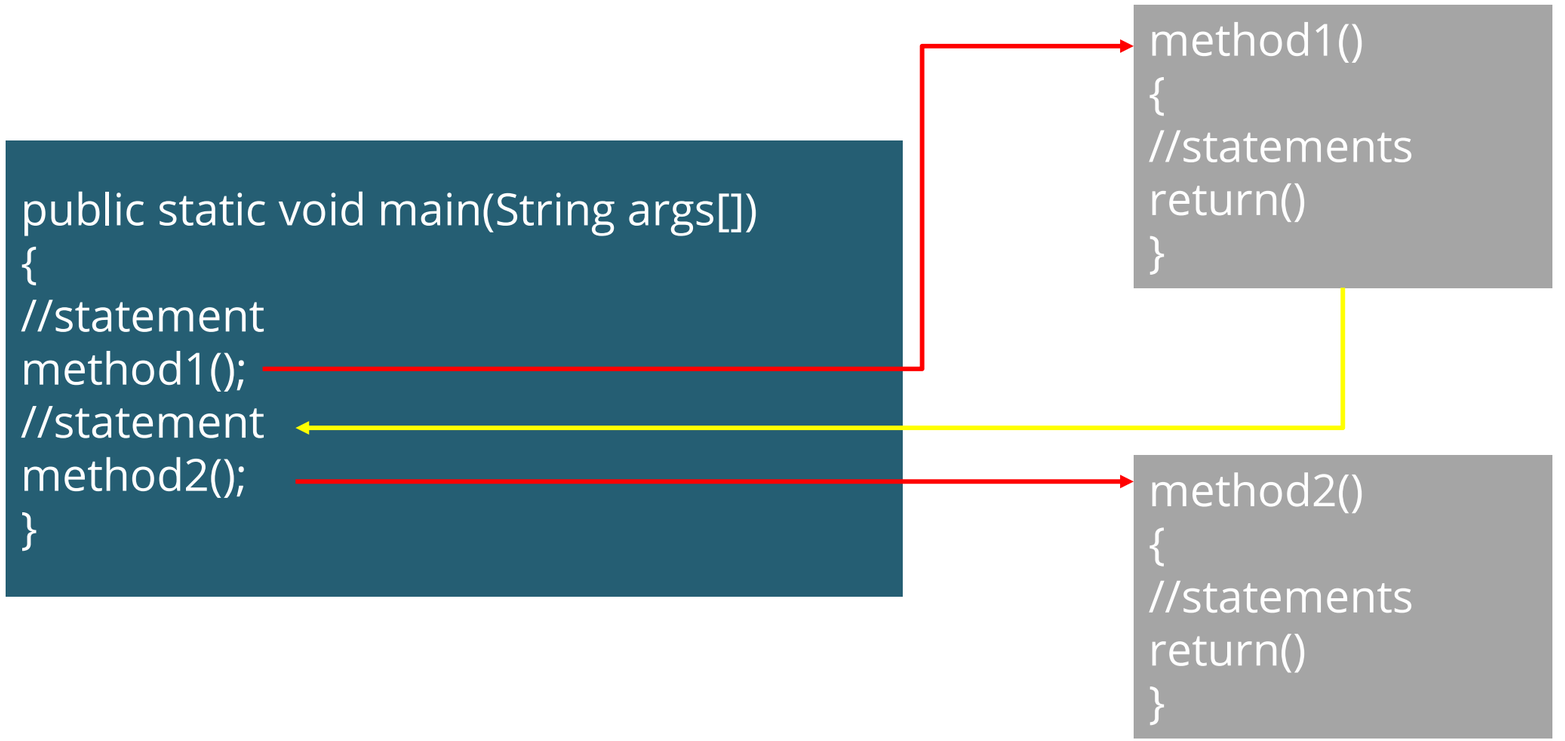
By the end of this lesson, you will be able to:

- Use Java methods
- Create and overload constructors
- Demonstrate collections
- Use inner classes in code
- Declare, initialize, and use single-dimensional as well as multidimensional arrays
- Demonstrate regular expressions

Methods

Methods

A method is a collection of statements, which performs specific tasks and returns results to the caller. It allows you to write reusable code and divide the program into several small units.



Return statement

- The *return* statement is a control flow statement, which terminates the execution of method and returns control to its caller.
- If a method has a *void* return type, it means the method does not return anything.

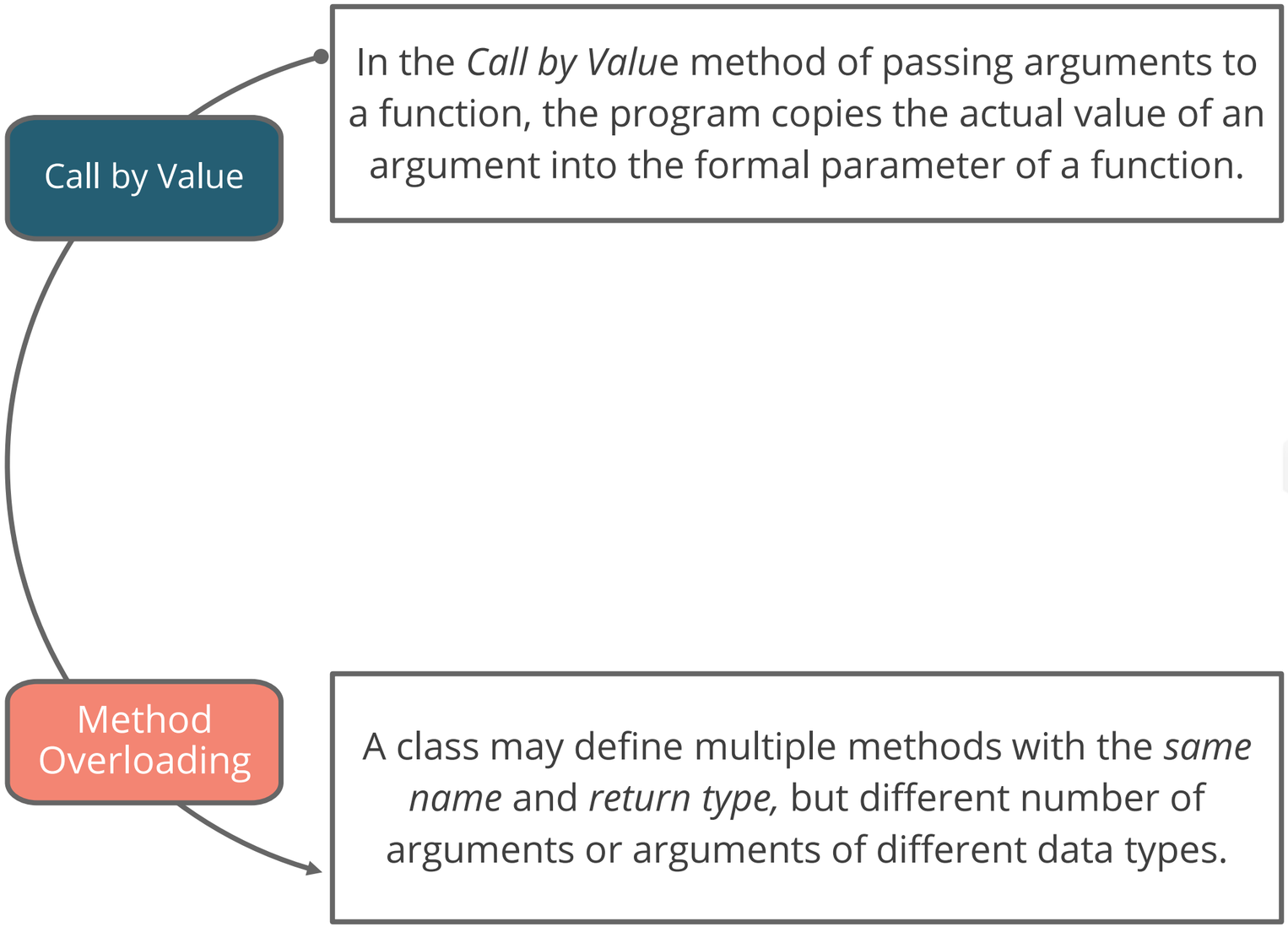
Return type is null as work() is not returning anything

```
public void work(){
System.out.println("Welcomes everyone");
}
public static void main(String[] args){
work();
}
```

Return type is int as welcome() returns i, which is of the type Integer

```
public class demoValue {
        public int welcome(int a,int b){
        int i=0;
        for(i=a;i<b;i++){
                System.out.println(i);}
        return i; }
public static void main(String[] args){
demoValue dv=new demoValue();
int num=dv.welcome(2, 5);
System.out.println(num);
}}
```

Calling a Method



Methods

Duration: 15 min.

Problem Statement:

Write a program to demonstrate different methods of different return types.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the methods:

1. Create a Java project in your IDE
2. Write a program in Java to create a method
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

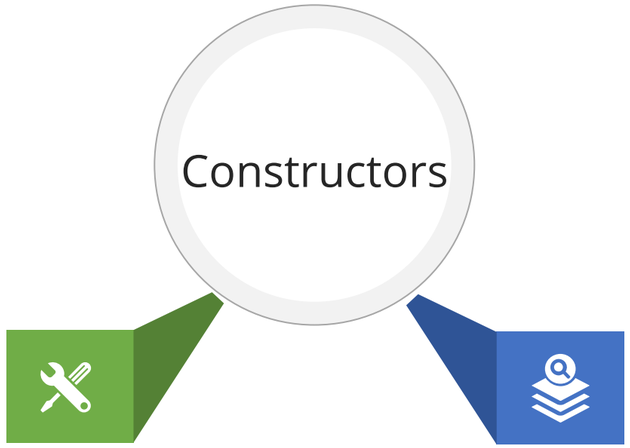
Constructors

Constructors

Constructors are used to initialize an object when it is created. It has the same name as the class but no explicit return type.

Syntax:

```
class ClassName{
ClassName()
 }
```



No Argument Constructor:

- It does not have any parameter.
- It is also called “Default” constructor.
- If constructor is not defined in a class, then compiler creates a **default constructor (with no arguments)** for the class.

Parameterized Constructor:

- It can have multiple parameters.
- To initialize fields of the class with own values, use parameterized constructor.

Difference between Constructor and Method

Constructor	Method
Constructor must not have the return type.	Method must have the return type.
Constructor name must be same as the class name.	Method name must not be same as the class name.
Constructor is used to initialize the state of an object.	Method is used to expose the behavior of an object.
Constructor is invoked implicitly.	Method is not invoked implicitly.

Constructors

Duration: 30 min.

Problem Statement:

Write a program to demonstrate the uses of constructors and its types.

ASSISTED PRACTICE

Assisted Practice: Guidelines

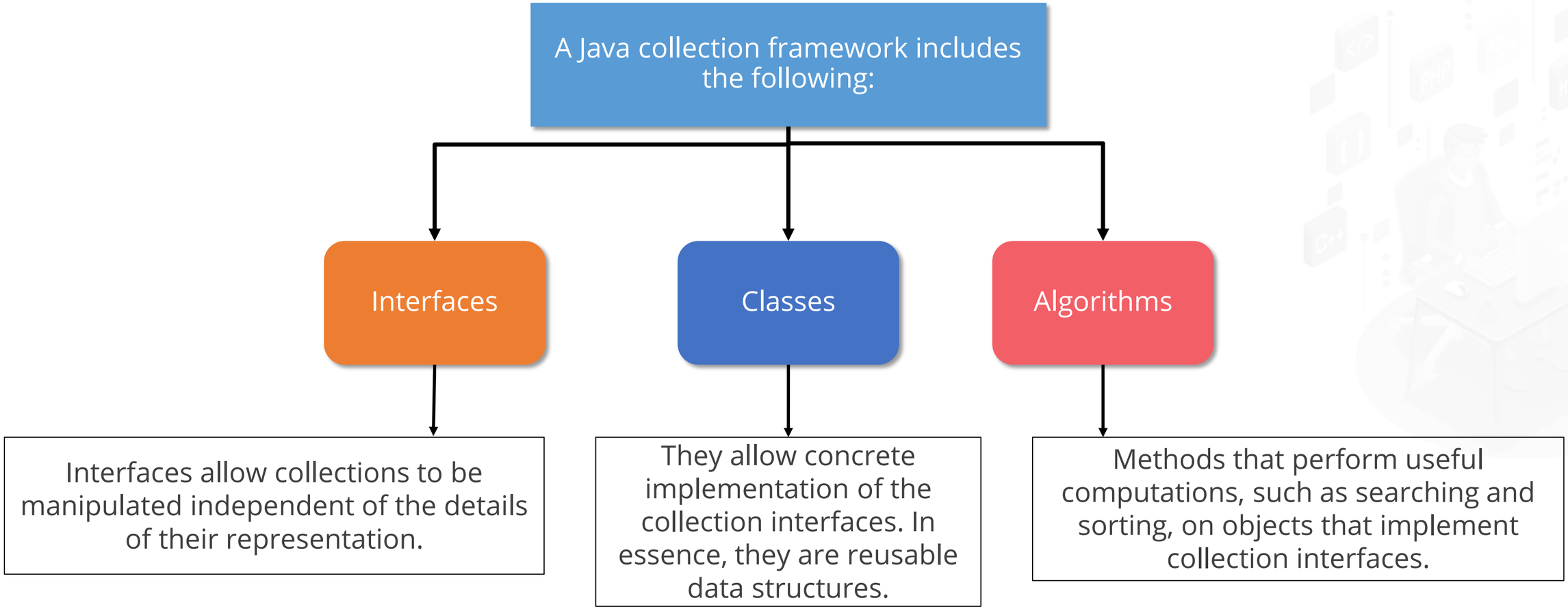
Steps to demonstrate the constructors:

1. Create a Java project in your IDE
2. Write a program in Java to create a constructor
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

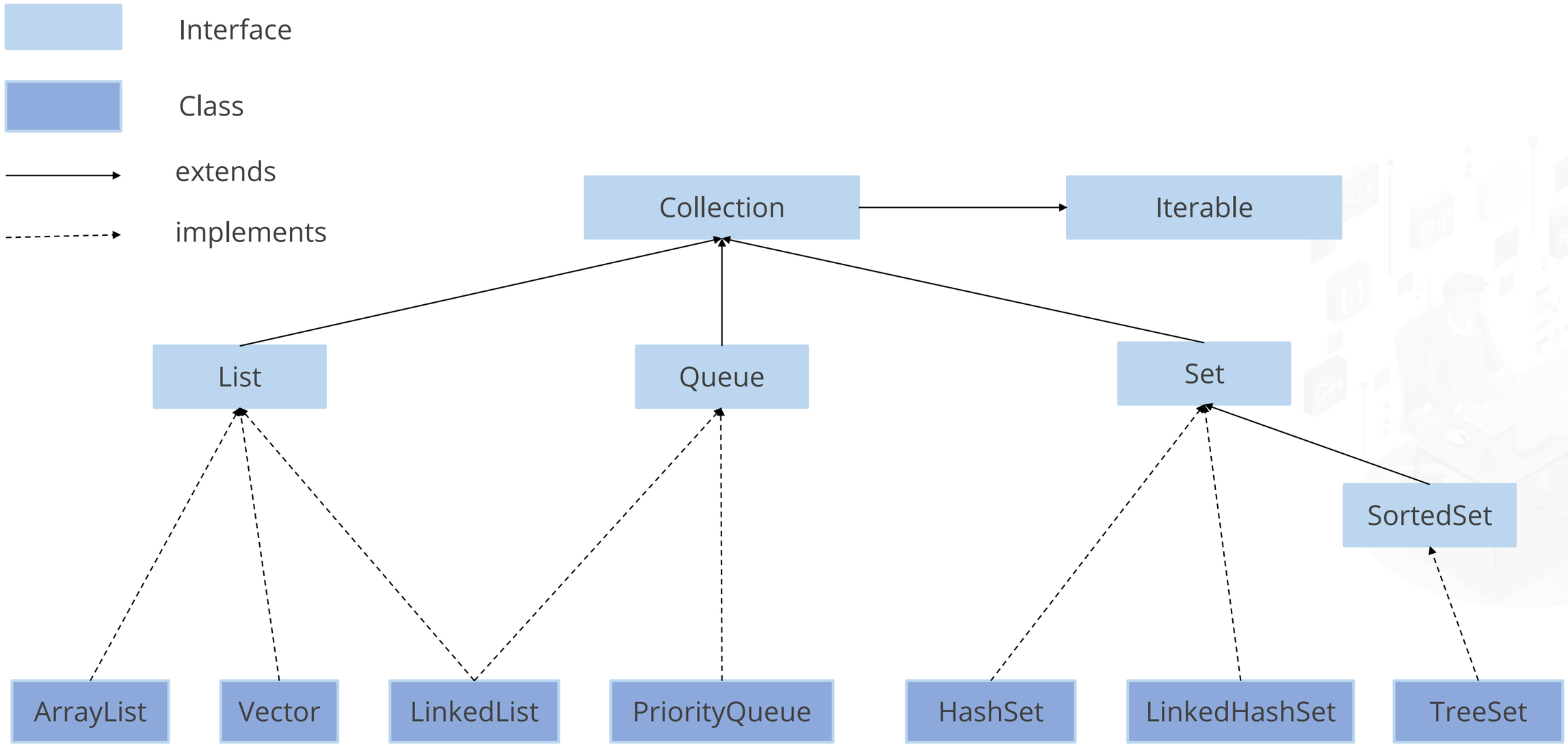
Collections

Collections

A Java *collection* framework provides an architecture to store and manipulate a group of objects.



Hierarchy of Collection Framework

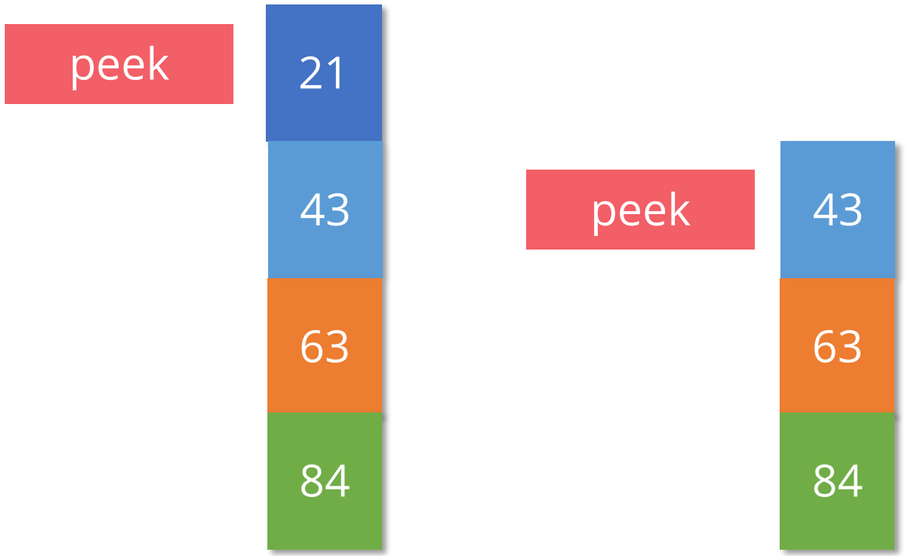


List

A list is an ordered collection of elements which may contain duplicates.
Lists are further classified into the following:

Queue

- Queue is a data structure which follows first in first out (FIFO) algorithm.
- Example: Remove one element from the Queue –



- A *PriorityQueue* class allows you to initialize a queue:

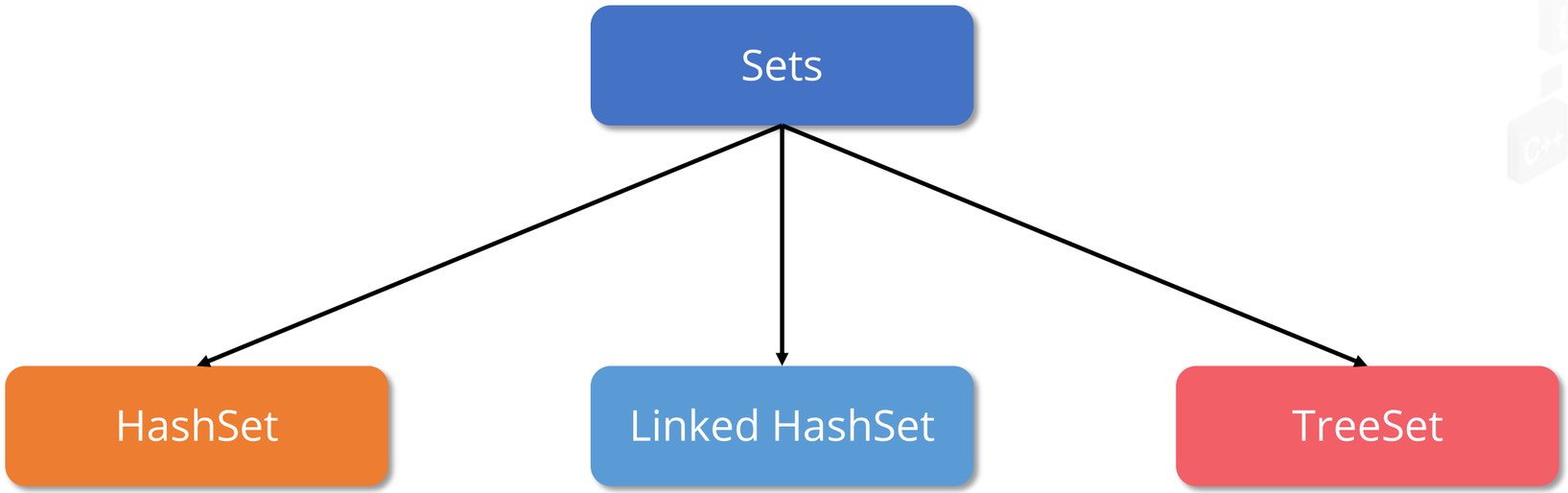
```
Queue B = new PriorityQueue();
```

- Another way to initialize queue :

```
Queue A = new LinkedList();
```

Set

- A set refers to a collection that cannot contain duplicate elements.
- It is mainly used to model the mathematical set abstraction.
- Set has its implementation in various classes such as HashSet, TreeSet, and Linked HashSet.



Collections

Duration: 30 min.

Problem Statement:

Write a program to demonstrate the uses of collections and its types.

ASSISTED PRACTICE

Assisted Practice: Guidelines

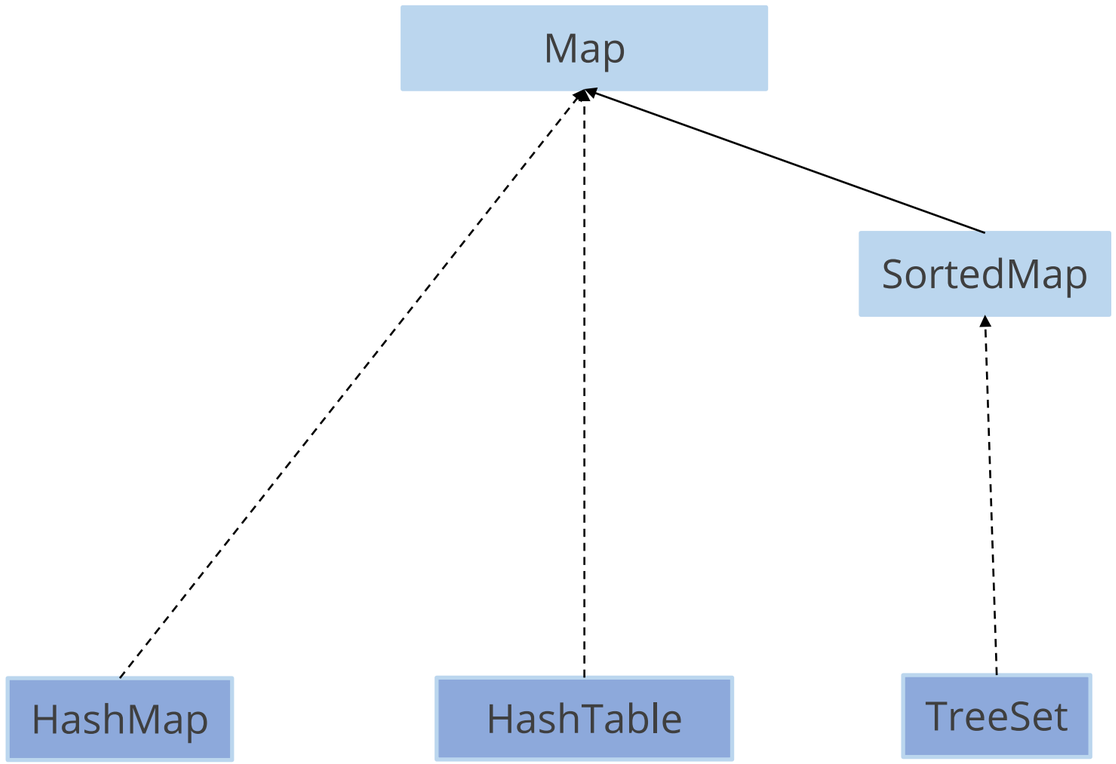
Steps to demonstrate the collections:

1. Create a Java project in your IDE
2. Write a program in Java to create a collection
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

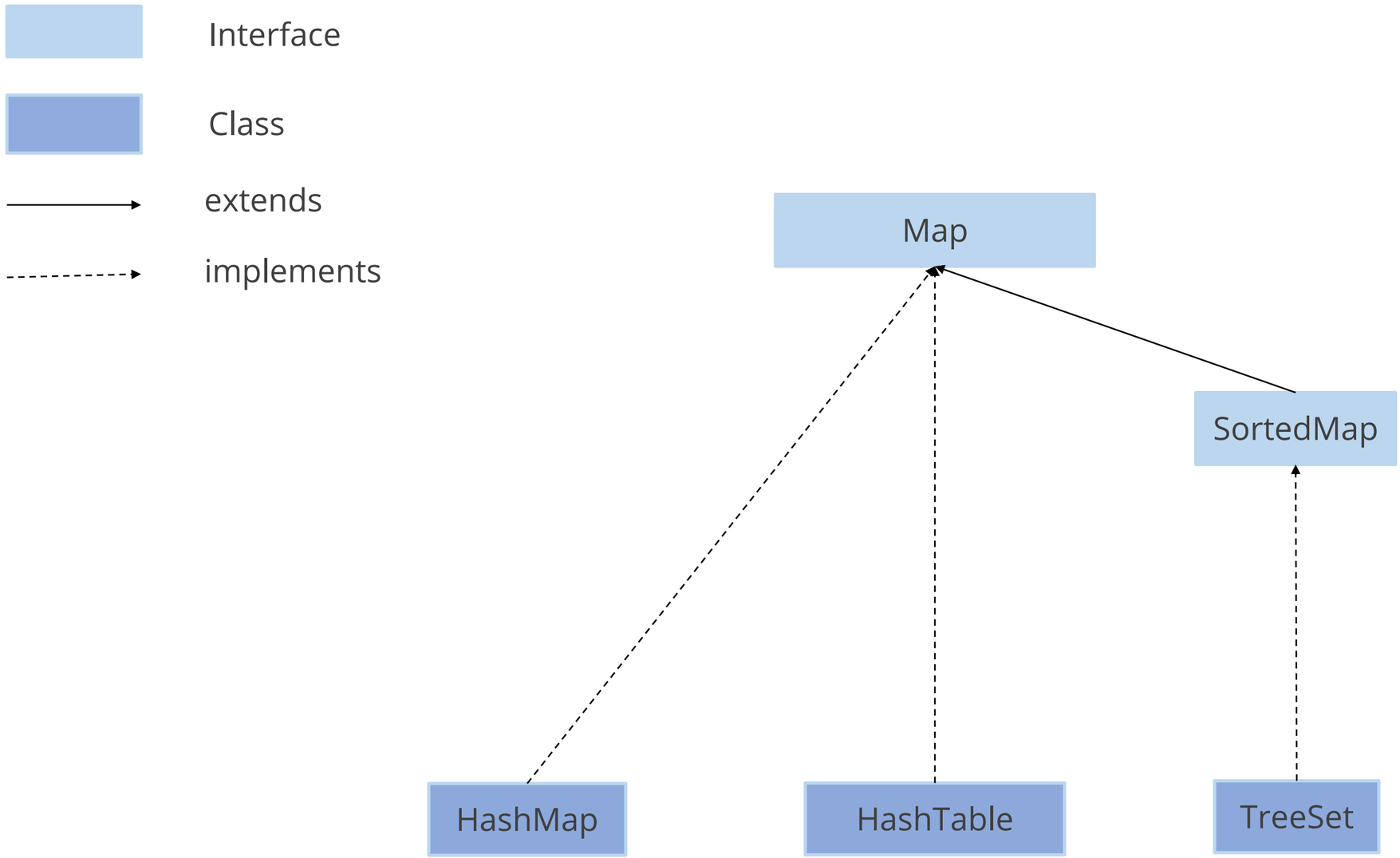
Map

Map

- The map interface maps ***unique keys*** to ***values.***
- It contains a key that is an object used to retrieve a value.
- A map object only stores ***key-value pairs.***



Hierarchy of Collection: Map



Map

Duration: 15 min.

Problem Statement:

Write a program to demonstrate the uses of Map.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the Map:

1. Create a Java project in your IDE
2. Write a program in Java to demonstrate working of Map
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Inner Classes

Inner Classes

An inner class is a class which is declared inside another class.

Java inner classes are used to achieve security mechanism.

Java inner classes are nothing but non-static nested classes.

Less code is required to write an inner class.

An inner class can access all the data members and member functions of outer class including private data members.

Java inner class can be declared as public, private, and protected.

Inner Classes

Duration: 15 min.

Problem Statement:

Write a program to demonstrate the working of inner classes.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the inner classes:

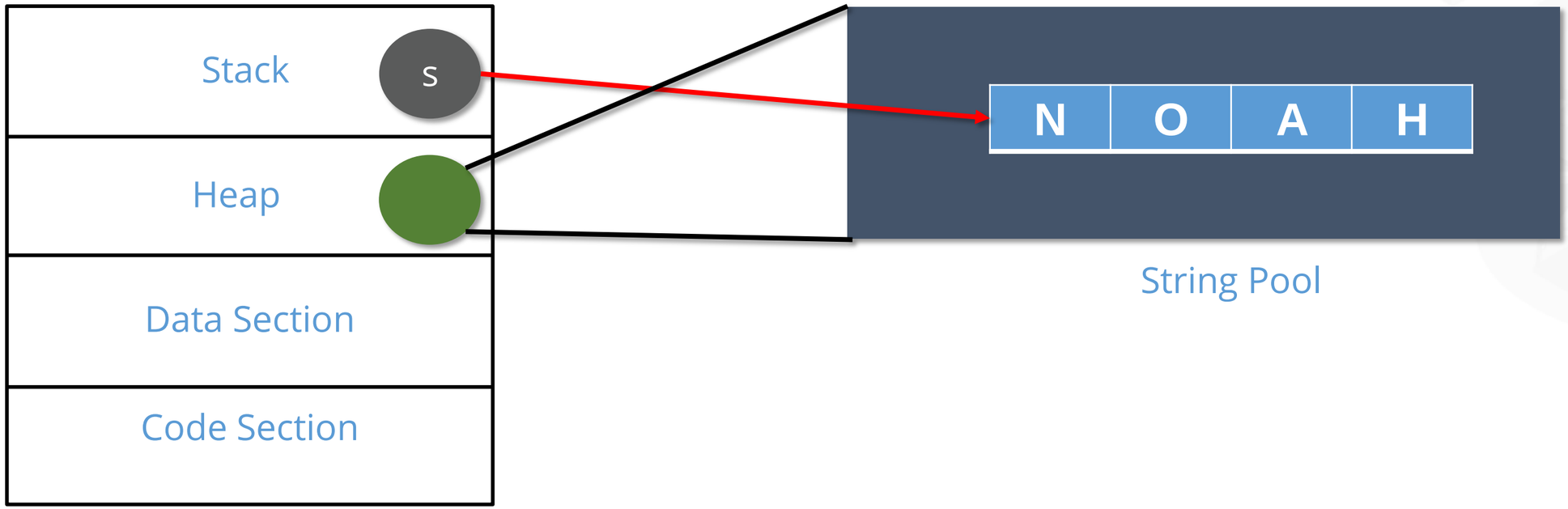
1. Create a Java project in your IDE
2. Write a program in Java to create an inner class
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Strings

Strings

Strings are objects that represent the sequence of characters. It cannot be changed once created. It is not a primitive data type.

Syntax	String s="NOAH"	String s=new String("NOAH")



StringBuffer and StringBuilder

StringBuffer

- StringBuffer is mutable. It means one can change the value of the object.
- The objects created using StringBuffer are stored in the heap.
- It is thread-safe.
- Its performance is affected due to its thread-safe property. Hence, StringBuffer is slower than StringBuilder.

StringBuilder

- StringBuilder is mutable like StringBuffer.
- It is not thread-safe.
- StringBuilder is faster than StringBuffer.

Syntax

```
StringBuffer sbr= new StringBuffer (“example1”);
```

Syntax

```
StringBuilder slr= new
StringBuilder(“example2”);
```

Strings

Duration: 30 min.

Problem Statement:

Write a program to create strings and display the conversion of string to StringBuffer and StringBuilder.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the Strings:

1. Create a Java project in your IDE
2. Write a program in Java to create a string, a StringBuffer and a StringBuilder
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Arrays

Arrays

An array is a data structure used to store elements of the same data type. It is index-based. The first element refers to index 0. You can create one dimensional as well as multidimensional arrays in Java.

Single-dimensional Arrays

In single-dimensional, arrays elements are stored in rows only.

Example

int a= new int[5];

a[0]	a[1]	a[2]	a[3]	a[4]

Multidimensional Arrays

In multidimensional arrays, elements are stored in the form of rows and columns.

Example

int s [ ] [ ]=new int [3] [3]

S[0][1]	S[0][1]	S[0][2]
S[1][0]	S[1][1]	S[1][2]
S[2][0]	S[2][0]	S[2][2]

Arrays

Duration: 30 min.

Problem Statement:

Write a program to create single-dimensional and multidimensional arrays.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the Arrays:

1. Create a Java project in your IDE
2. Write a program in Java to create an array
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Regular Expressions

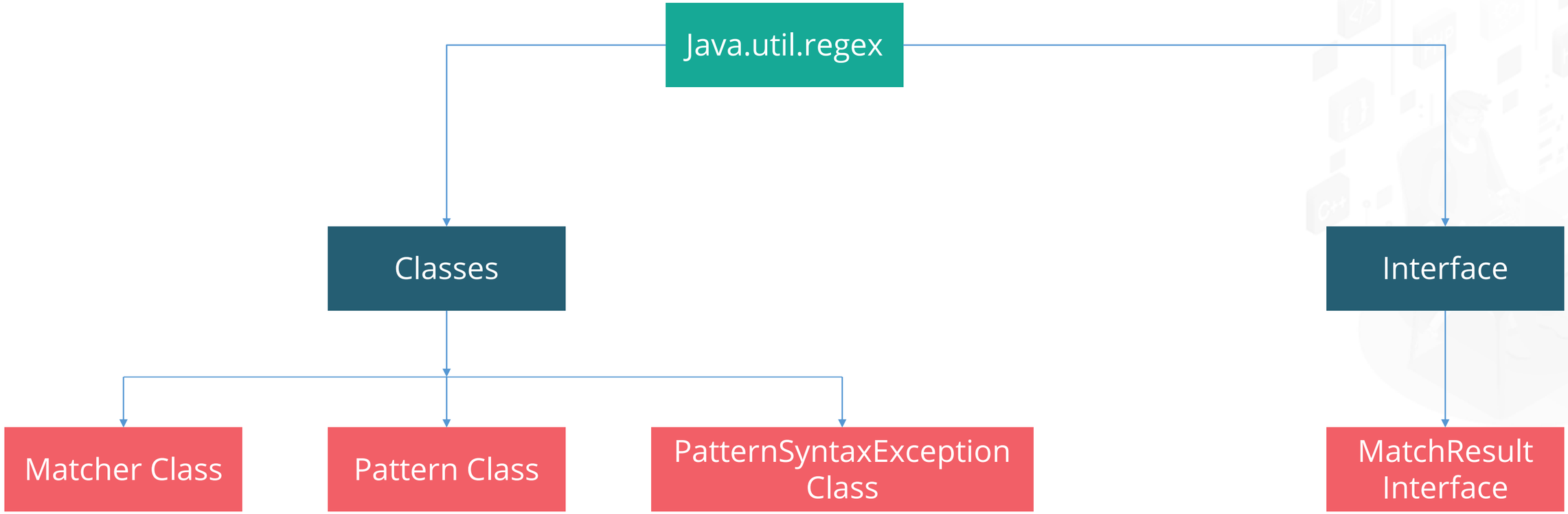
Regular Expressions

A regular expression is an API used to define a pattern for searching or manipulating strings.

Regular expression is widely used to define the constraint on strings like password and email validation.

Regular Expressions

For regular expressions, the java.util.regex package provides three classes and one interface.



Patterns

Here are some matching patterns used in a regular expression.

Patterns	Description
abc	This sequence should be followed exactly
[abc]	Any one letter from a, b, or c should match
[^abc]	Any letter other than these three should match
[a-z]	Any letter from a to z can be present in the sequence
[a-zA-z0-9]	Any letter from a to z or A to Z or any digit from 0 to 9 can be in the sequence
.	Any character other than the line terminator can be in the sequence
^	To check if any character is present at the beginning of a line
$	To check if any character is present at the end of a line
\b	To verify if any character is present at the word boundary or not
\B	To verify that no character is present at the word boundary
\G	To check that the character is present at the end of the previous match

Meta Characters

The meta-characters have predefined meanings and are used to make easier use of patterns.

Patterns	Description
\d	A digit [0-9]
\D	A non-digit[^0-9]
\s	A whitespace character
\S	A non-whitespace character
\w	A word character
\W	A non-word character[^\w]

Quantifiers

A quantifier defines how often an element can occur.

Patterns	Description
*	Occurs zero or more times
+	Occurs one or more times
?	Occurs no or one time
{X}	Occurs X number of times
{X,Y}	Occurs between X and Y times
*?	It tries to find the smallest match. This makes the regular expression stop at the first match

Regular Expressions

Duration: 20 min.

Problem Statement:

Write a program to search a specific string from the given set of strings using regular expressions.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the Regular Expressions:

1. Create a Java project in your IDE
2. Write a program in Java to create a regular expression
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Key Takeaways

- A Java method is a collection of statements that are grouped together to perform a specific operation.
- A Java constructor resembles an instance method, but it does not have a return type.
- Collection framework includes interfaces, classes, and algorithms.
- Inner classes are used to logically group classes and interfaces in one place to make them more readable and maintainable.
- Regular expression in an API uses patterns to search and manipulate strings.

Validation of an Email ID

Duration: 30 min.

Problem Statement:
Write a program to search a string entered by a user from the array of strings.

LESSON-END PROJECT

Before the Next Class

Course: Core Java

You should be able to:

- Use thread class and demonstrate multithreading
- Explain polymorphism and inheritance
- Explain the types of exceptions
- Explain the advantages of exception handling
- Create a try-catch block and demonstrate exception handling
- Demonstrate file handling
- Explain serialization and deserialization